SSF-NET: A SPATIAL–SPECTRAL FUSION NETWORK FOR ACCURATE PDE OPERATOR LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine learning methods for solving partial differential equations (PDEs) significantly improve computational efficiency over traditional numerical solvers such as finite element method (FEM). However, PDE dynamics inherently span both local and global dynamics: disturbances propagate and decay gradually through the domain, while global coupling effects can influence the whole system simultaneously. Capturing both types of dynamics with a low complexity model remains a key challenge. To address this, we propose the Spatial-Spectral Fusion Network (SSF-Net), built around a dual-branch Spatial–Spectral Fusion (SSF) block comprising a spatial branch and a spectral branch. The spatial branch employs a multi-directional Mamba, which mitigates its inherent causal, single-direction scanning bias and capture local interactions with reduced directional bias at linear complexity. The spectral branch leverages a Fourier transform with a block-diagonal MLP to efficiently capture global spectral features with reduced parameter count and complexity. The SSF block is embedded in a multiresolution convolutional encoder-decoder, which extracts multi-scale features. We evaluate SSF-Net on a comprehensive PDE benchmark spanning multiple task types and assess its long-horizon stability. Compared with attention-based and convolution-based baselines, SSF-Net achieves lower prediction error with fewer parameters while retaining near-linear asymptotic complexity in the number of spatial tokens.

1 INTRODUCTION

Scientific and engineering problems such as aerodynamics and wave propagation involve solving complex partial differential equations (PDEs). Conventional PDE solvers such as the finite element method (FEM), the finite volume method (FVM), and spectral methods discretize the continuous physical domain into a finite set of points, cells, or elements (Quarteroni & Valli, 1994). These methods are accurate and physically grounded, but remain computationally expensive at scale. This challenge has motivated the development of data-driven machine learning methods for solving PDEs, among which neural operators have emerged as a particularly promising direction.

Neural operators aim to learn the underlying PDE solution operator, which directly maps an input function space—including initial and boundary conditions, coefficients, etc.—to the corresponding PDE solution (Li, 2021). Compared to conventional simulation, these methods achieve lower computational cost and faster inference (Takamoto et al., 2022). Moreover, because neural operators are data-driven, they remain applicable even when the governing equations are incomplete or unknown. Existing neural operators are built on a variety of backbones, including convolutions (Raonic et al., 2023), graph neural networks (Li et al., 2020b), Fourier transforms (Li et al., 2020a; 2023a;b), and transformers (Wu et al., 2024; Luo et al., 2025; Herde et al., 2024; Holzschuh et al., 2025b).

Nevertheless, PDE dynamics inherently span both local and global scales. Locally, disturbances such as shock fronts, boundary layers, and localized stress concentrations propagate gradually, decay with distance, and are primarily governed by neighboring interactions. Globally, coupling effects—such as pressure fields in incompressible flow or potential fields in Poisson-type equations—can instantaneously influence the entire system, independent of spatial distance.

Existing methods struggle to efficiently integrate local and global dynamics. Firstly, convolution- and graph-based models typically require multiple layers to capture long-range dependencies. Fourier-based models can lose fine-scale spatial details through mode truncation. Transformer-based

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

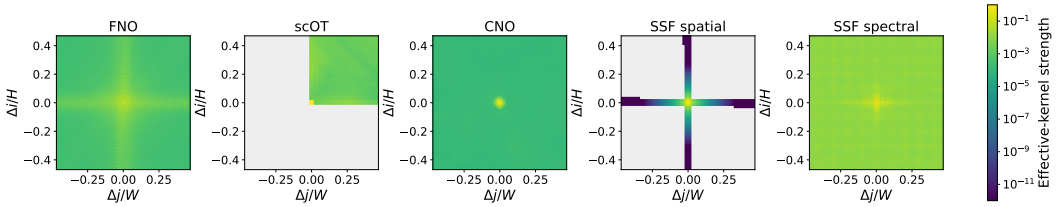

Figure 1: Effective-kernel strength on FNS-KF. Effective-kernel strength is a measure of how strongly infinitesimal changes in the source features affect the query output. See Section 4.4 and Appendix F for details. Axes denote normalized spatial displacements. All panels share a logarithmic color scale, with gray indicating zero.

models face a trade-off between the quadratic cost of global attention and the restricted receptive fields of window-based attention. Moreover, these methods do not explicitly model decaying local interaction. Figure 1 illustrates how input locations across the spatial domain influence the output at a given query position. CNO exhibits domain-wide responses concentrated near the query, while FNO shows broad spatial coverage with enhanced influence along the horizontal and vertical axes through the query. The scOT block is restricted to its attention window. None of them shows an explicit gradual decaying influence with distance which is common in the local interaction of PDEs.

To address these gaps, we introduce SSF-Net, a dual-branch neural operator that jointly models local and global PDE dynamics at near-linear complexity. The spectral branch extracts global dynamics via a Fourier transform, while the spatial branch extracts local dynamics via a state-space model (SSM). This design combines complementary local and global representations and achieves improved prediction accuracy with fewer parameters than the compared baselines. As shown in Figure 1, the SSF spatial branch exhibits a concentrated, directional response that rapidly weakens away from the query, while its spectral branch provides broad spatial coverage. These complementary patterns support the integration of local sensitivity and global interaction within PDEs. We make the following contributions:

- We propose a novel dual-branch architecture that combines SSM (Gu & Dao, 2023; Tiwari et al., 2025) for local dynamic extraction with a Fourier transform for global dynamic extraction. The resulting model, termed the Spatial-Spectral Fusion Network (SSF-Net), is embedded within a U-Net-style multiresolution encoder-decoder to extract multi-scale features while reducing parameter count.
- We show that SSF-Net has near-linear asymptotic complexity in the number of spatial tokens, avoiding the quadratic token-mixing cost of full self-attention.
- We evaluate SSF-Net on a diverse collection of PDE benchmarks and show that it consistently achieves lower prediction errors with fewer parameters than the compared baselines.

2 RELATED WORK

Neural operators learn mappings between function spaces to approximate PDE solution operators (Li, 2021). Liu et al. (2025) classify the neural operators into three categories: deep operator networks (DeepONets), integral kernel operators, and transformer-based neural operators.

DeepONet combines branch-network features of the input function with trunk-network features of query coordinates through an inner product to evaluate nonlinear operators at arbitrary locations (Lu et al., 2021). Its original architecture does not explicitly separate local and global interactions.

Integral kernel operators approximate the PDE solution function via parameterized kernels. FNO and its variants (Li et al., 2020a; Guibas et al., 2021; Li & Salim, 2025) implement the kernel operator in the Fourier domain. These methods model global interactions through Fourier-domain processing. Geo-FNO (Li et al., 2023a), GINO (Li et al., 2023b) and U-FNO (Wen et al., 2022) generalize the method to irregular geometries and multiphase flow. In contrast, GNO (Li et al., 2020b) and CNO (Raonic et al., 2023) instead parameterize the kernel using GNNs or CNNs, respectively, to model local interactions through spatial neighborhood aggregation.

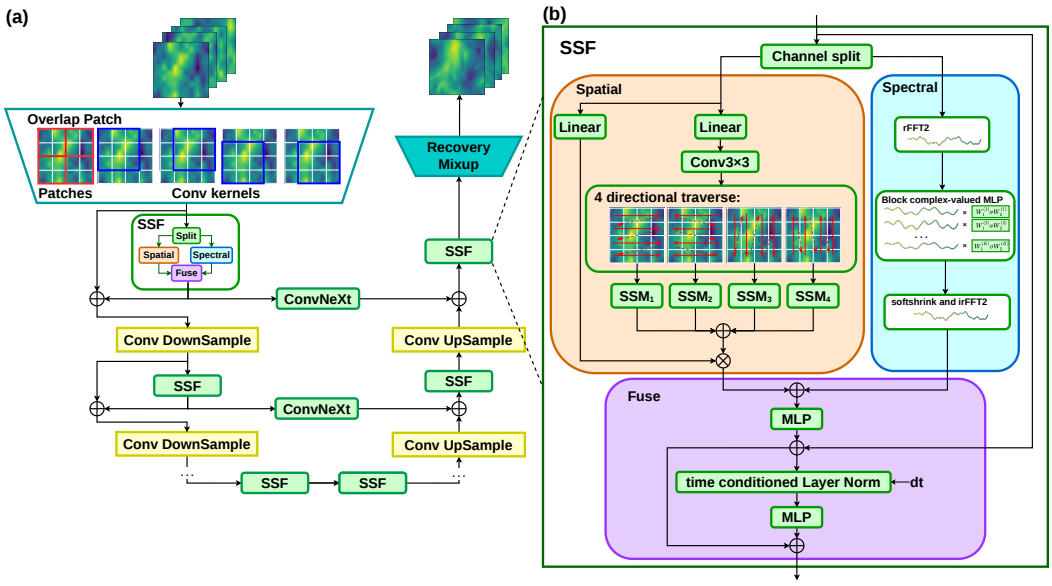

Figure 2: Overview of SSF-Net. (a) U-Net style encoder–decoder with ConvNeXt blocks and overlapping patch embedding. (b) Two parallel branches: a spatial branch with four-directional Mamba scans, and a spectral branch with a rFFT2, a block-diagonal MLP, and soft-shrinkage.

Transformer-based neural operators discretize the continuous domain, treating each grid point as a token, and use attention as the kernel (Wu et al., 2024; Luo et al., 2025; Serrano et al., 2024; Wang et al., 2024). Transformer-based operators are also widely used in multi-task foundation models(Herde et al., 2024; Holzschuh et al., 2025b;a; McCabe et al., 2023). Full self-attention permits direct interactions across the domain through all-to-all token aggregation. However, the quadratic complexity limits its training efficiency. Window-based attention, such as SWIN (Liu et al., 2021; Herde et al., 2024; Holzschuh et al., 2025b), mitigates this cost by restricting attention to local windows, achieving linear complexity. Interactions beyond an individual window generally require propagation through shifted windows, stacked layers, or the multiscale hierarchy.

Beyond this taxonomy, SSMs have recently emerged as a fourth paradigm for neural operators, leveraging recurrent state updates to achieve linear-complexity sequence modeling (Zheng et al., 2024; Tiwari et al., 2025). Although SSMs can model long-range interactions through recurrent state propagation, distant positions communicate through intervening state updates. Contractive transitions introduce a scan-distance-dependent attenuation bias, while Mamba's selective mechanism makes interactions input-adaptive.

These architectures provide different mechanisms for spatial interaction, with distinct receptive-field, adaptivity, and computational properties. This motivates SSF-Net, which explicitly fuses spatial and spectral branches to jointly model local and global interactions at near-linear complexity.

3 METHOD

We propose SSF-Net to jointly model local and global interactions in PDE solution operators with near-linear asymptotic complexity. For time-dependent problems, predictions are generated through autoregressive rollout. The formal problem setting and notation are provided in Appendix A.

3.1 OVERALL ARCHITECTURE

As shown in Figure 2(a), SSF-Net adopts a multiresolution encoder-decoder structure, with a dual-branch Spatial-Spectral Fusion (SSF) block introduced at each resolution scale. Within each SSF block, the input is split into two parts, processed independently by the spatial and spectral branches, and then merged (Figure 2(b)). The SSM-based spatial branch emphasizes local interactions, while

the multi-directional traverse mitigates the inherent causal bias of SSMs arising from their sequential processing of the input. The FFT-based spectral branch transforms the 2D input into the spectral domain, where nonlinearity is introduced to enhance the model's expressiveness, and a block-diagonal MLP reduces computational cost. Downsampling and upsampling across resolutions are implemented via convolution, and the overlapping patching preserves local continuity across neighboring patches by allowing adjacent patches to share spatial information.

SSF offers the following advantages. (1) The split-process-fuse structure avoids channel inflation, thereby reducing computational cost. (2) At each scale, the spectral branch in SSF achieves a global receptive field without relying on deep stacking or large kernels, as required by convolution-based models (Raonic et al., 2023), or long-range sequential propagation, as required by SSM-based models (Tiwari et al., 2025; Gu & Dao, 2023). This enables direct global interactions at each resolution, alongside input-adaptive local interactions in the spatial branch.

3.2 OVERLAPPING PATCH EMBEDDING

Neighboring points in PDE data exhibit strong spatial correlations. Instead of partitioning the feature map into disjoint patches, overlapping patch embedding employs a convolutional kernel larger than the stride, allowing neighboring patches to share overlapping receptive fields (Wang et al., 2022). This design facilitates local interactions, preserves spatial continuity across patch boundaries, and alleviates information loss during patching. Specifically, adjacent patch windows are enlarged to overlap by o pixels. Given an input $X_{\text{in}} \in \mathbb{R}^{H\times W\times C}$ and a target patch size p, we apply a convolution with kernel size $p + o$, stride p and C' output channels, producing

$$X_{\text{patch}} = \text{LN}(\text{Conv}(X_{\text{in}})), \qquad X_{\text{patch}} \in \mathbb{R}^{(H/p)\times(W/p)\times C'}, \tag{1}$$

where LN denotes the time conditioned layer normalization defined in Herde et al. (2024). See Appendix D.1 for details of overlapping patch, and Appendix D.3 for depatch.

3.3 SPATIAL-SPECTRAL FUSION (SSF)

For notational simplicity, we reuse H, W, C to denote the spatial and channel dimensions of the input at the current resolution stage throughout Section 3. Their concrete values differ across resolution levels and are updated after each downsampling or upsampling operation.

Feature split and fusion. To avoid channel inflation caused by the dual-branch structure, the input $X \in \mathbb{R}^{H\times W\times C}$ is split along the channel dimension into two parts, processed separately by the spatial and spectral branches. The outputs of the two branches are then fused (Figure 2(b)):

$$X_s, X_f = \text{split}(X), \qquad X_s, X_f \in \mathbb{R}^{H\times W\times D}, \tag{2}$$

$$X_{\text{local}} = f_{\text{spatial}}(X_s), \qquad X_{\text{global}} = f_{\text{spectral}}(X_f), \tag{3}$$

$$X_{\text{out}} = \text{MLP}\big(\text{LN}(X_{\text{local}} + X_{\text{global}})\big), \tag{4}$$

where $D = C/2$. f_{spatial} and f_{spectral} denote the spatial and spectral branches, detailed in the following paragraphs. LN denotes the time conditioned layer normalization defined in Herde et al. (2024) over the channel dimension and $\text{MLP} : \mathbb{R}^D \to \mathbb{R}^C$ acts pointwise, lifting the fused features back to the block width. X_{out} is the output of the SSF block, which is combined with a residual connection before being passed to the downsampling/upsampling operator. See Appendix D.2 for details of MLP.

Multi-directional Mamba. We now detail the spatial branch f_{spatial} introduced in equation 3. Following the scan patterns in Teng et al. (2024) and Liu et al. (2024), we employ a four-directional Mamba traversal as our spatial branch to emphasize local interactions. SSMs are originally designed for sequential data with a causal structure, where each step depends only on the preceding tokens. Spatial interactions in PDE fields need not follow a causal scan ordering and can occur in all directions. To bridge this gap, we introduce a multi-directional traverse mechanism. As shown in the spatial branch in Figure 2(b), the input is scanned along four paths: top-left to bottom-right, bottom-right to top-left, top-right to bottom-left, and bottom-left to top-right. Each traversal follows a raster

ordering (each row or column is scanned in a consistent direction). Each path defines an ordering π_n that flattens the 2D grid into a 1D sequence:

$$X_n = \pi_n\big(\mathrm{Conv}(\mathrm{Linear}(X_s))\big), \qquad n = 1, 2, 3, 4, \tag{5}$$

where the linear projects $X_s \in \mathbb{R}^{N\times D},\ N = H \times W$ to $X_n \in \mathbb{R}^{N\times D_E},\ D_E = E \cdot D$, where E is the expansion factor in Mamba. The four sequences contain identical elements and differ only in their ordering. See Appendix D.4 for details of multi-directional traverse.

To better exploit the four traversal sequences, we adopt a multi-head SSM similar to multi-head attention (Vaswani et al., 2017). The channel dimension D_E of each sequence is split into M heads, each of dimensionality D_E/M stacking across the four traversal directions yields $4M$ heads in total. Within each direction, the outputs of all heads are concatenated, and the resulting representations from the four directions are summed:

$$X_{\mathrm{ssm}} = \sum_{n=1}^{4} \pi_n^{-1}\Big(\mathrm{Concat}\big(\mathrm{SSM}(X_n^1), \ldots, \mathrm{SSM}(X_n^M)\big)\Big), \tag{6}$$

$$X_{\mathrm{local}} = \mathrm{Linear}\big(\sigma(X_{\mathrm{ssm}}) \otimes \sigma(\mathrm{Linear}(X_s))\big), \tag{7}$$

where n indexes the four traversal directions, and $X_n^h \in \mathbb{R}^{N\times D_E/M}$ denotes the h-th head ($h = 1, \ldots, M$) of the n-th traversal sequence X_n. π_n^{-1} restores each output to the original grid layout so that the four directions are spatially aligned before summation. We use GeLU (Hendrycks & Gimpel, 2016) as the activation function σ. The output $X_{\mathrm{local}} \in \mathbb{R}^{H\times W\times D}$ is the aggregated spatial feature, passed to the subsequent fusion step. See Appendix B for underlying state-space formulation and Appendix D.4 for implementation details of Mamba.

Adaptive FNO. Following AFNO (Guibas et al., 2021), the spectral branch mixes tokens in the frequency domain. Given the spectral input X_f, we first apply a 2D FFT:

$$Z = \mathrm{RFFT2D}(X_f), \qquad Z \in \mathbb{C}^{H\times W_r\times D}, \quad W_r = \lfloor W/2 \rfloor + 1, \tag{8}$$

To reduce the parameter count of channel mixing, the channel dimension is partitioned into K blocks, and a two-layer complex-valued MLP is applied independently within each block and shared across all tokens:

$$\tilde{Z}^{(\ell)} = W_2^{(\ell)}\,\sigma\big(W_1^{(\ell)} Z^{(\ell)} + b_1^{(\ell)}\big) + b_2^{(\ell)}, \qquad \ell = 1, \ldots, K, \tag{9}$$

where $W_1^{(\ell)}, W_2^{(\ell)} \in \mathbb{C}^{D/K\times D/K}$ and σ denotes ReLU applied to the real and imaginary parts separately. Since frequency modes are sparse, with most energy concentrated in low frequencies, we sparsify $\tilde{Z}$ via soft-shrinkage:

$$\hat{Z} = S_\lambda(\tilde{Z}), \qquad S_\lambda(x) = \mathrm{sign}(x)\max\{|x| - \lambda, 0\}, \tag{10}$$

where λ is a fixed tuning parameter that controls the sparsity. After transforming back to the spatial domain via an inverse FFT, a residual connection preserves the branch input alongside the learned global interactions:

$$X_{\mathrm{global}} = \mathrm{IRFFT2D}(\hat{Z}) + X_f. \tag{11}$$

See Appendix D.5 for details of nonlinearity on complex domain.

An operator-learning interpretation of the two branches and their fusion is provided in Appendix C.

3.4 MULTIRESOLUTION ENCODER-DECODER

To extract multi-scale features and reduce the number of high-resolution spatial tokens processed at deeper stages, SSF-Net progressively downsamples the feature map across multiple resolution levels in the encoder, and symmetrically upsamples it in the decoder (Figure 2(a)). See Appendix D.2 for details.

Both downsampling and upsampling are implemented via convolution rather than patch merging (Liu et al., 2021): downsampling uses a strided convolution that halves the spatial resolution while doubling the channel dimension, and upsampling uses a symmetric transposed convolution that doubles the spatial resolution while halving the channel dimension. This convolutional design provides a learnable, spatially continuous aggregation that better preserves local smoothness in PDE fields, in contrast to patch merging, which performs a fixed spatial rearrangement followed by a linear projection.

3.5 COMPLEXITY ANALYSIS

We analyze complexity in terms of the spatial token count $N = HW$ and channel dimension D.

Spatial branch. The four-directional Mamba scans cost $O(NDL)$ (Gu & Dao, 2023; Gu et al., 2021), where L is the SSM state dimension. Linear projections cost $O(ND^2)$, while depthwise convolution and element-wise multiplication each cost $O(ND)$. The total is $O(N(DL + D^2))$, linear in N.

Spectral branch. The 2D FFT costs $O(ND \log N)$. Block-diagonal channel mixing reduces the MLP cost from $O(ND^2)$ to $O(ND^2/K)$ (Guibas et al., 2021), giving $O(ND \log N + ND^2/K)$ overall, near-linear in N.

Overall. Both branches scale linearly or near-linearly with N. The overlap patching, downsample/upsample convolution and convolutional skip modules all scale linearly in N and do not change the scaling. SSF-Net therefore retains a global receptive field with near-linear complexity, avoiding the $O(N^2)$ token-mixing cost of full self-attention. This does not guarantee faster execution: directional scans, spectral transforms, and multiresolution operations incur implementation- and hardware-dependent overhead.

4 EXPERIMENTS

We assess SSF-Net on a comprehensive dataset against several neural operator baselines in this section. And we also validate the contribution of major components of our model through ablation study.

4.1 IMPLEMENTATION DETAILS

Benchmarks. We evaluate SSF-Net on twelve two-dimensional PDE benchmarks. From PDE GYM (Herde et al., 2024), we include four incompressible flow tasks: NS-PwC tests the transport of discontinuous vorticity, NS-SL captures shear-layer instability and vortex roll-up, NS-Tracer-PwC examines coupled velocity and passive-tracer transport, and FNS-KF tests multiscale flow dynamics under sustained forcing. We further include Poisson-Gauss, CE-RP, Wave-Gauss, and ACE, covering elliptic problems, compressible flow, wave propagation, and reaction–diffusion dynamics. To assess long-horizon prediction, we use four APEBench-generated datasets (Koehler et al., 2024): Kolm-Smooth and Kolm-Turb for forced Kolmogorov flow, and KS-Weak and KS-Strong for Kuramoto–Sivashinsky dynamics, each evaluated over 100 autoregressive steps. Dataset details are provided in Appendix E.1.

Baselines We evaluate SSF-Net against following baselines: FNO (Li et al., 2020a), CNO (Raonic et al., 2023), PDE-Transformer (MC) (Holzschuh et al., 2025b), scOT (Herde et al., 2024) and Drift-Net (Li & Salim, 2025).

Training and metrics. We train a one-step operator with the relative L_1 loss, using the frequency-weighted loss of Li & Salim (2025) for Drift-Net, and evaluate time-dependent tasks through autoregressive rollout. We report three relative L_1 metrics: $\mathcal{E}_{\text{step}}$, which averages errors equally across forecast steps, $\mathcal{E}_{\text{full}}$, which measures the total trajectory error relative to the total reference magnitude, and $\mathcal{E}_{\text{final}}$, which evaluates the last forecast step. For multiple quantities of interest, errors are computed separately and averaged with equal weight. Formal definitions are provided in Appendix E.2.

4.2 MAIN RESULTS

Overall performance. As shown in Table 1, SSF-Net achieves the best performance on all twelve benchmarks. On the eleven time-dependent tasks, it consistently obtains the lowest step-averaged, full-rollout, and final-step relative L_1 errors. Compared with the second-best result for each task,

Table 1: Test errors on the PDE benchmarks. For time-dependent (TD) tasks, each entry reports $\mathcal{E}_{\text{step}}/\mathcal{E}_{\text{full}}/\mathcal{E}_{\text{final}}$. For the time-independent (TI) Poisson-Gauss task, only a single relative L_1 error is reported. All values are multiplied by 100. Lower is better. The best and second-best results in each row are shown in bold and underlined, respectively.

Dataset	Type	FNO	CNO	PDE-Transformer	scOT	Drift-Net	SSF-Net
Params	–	36M	18M	33M	20M	17M	11M
Poisson-Gauss	TI	14.37	1.16	0.18	0.42	0.46	**0.16**
NS-PwC	TD	0.45/0.45/0.68	2.25/2.25/4.58	0.31/0.31/0.42	1.24/1.24/1.90	2.88/2.88/4.74	**0.10/0.10/0.17**
NS-Tracer-PwC	TD	1.00/1.00/1.85	2.55/2.53/6.54	0.69/0.68/1.12	1.73/1.72/3.32	4.17/4.14/8.78	**0.18/0.18/0.33**
NS-SL	TD	0.68/0.69/1.24	3.23/3.32/7.11	0.30/0.31/0.56	1.90/1.94/3.59	2.29/2.34/4.45	**0.23/0.24/0.48**
FNS-KF	TD	0.41/0.41/0.63	2.33/2.34/4.80	0.28/0.29/0.43	1.37/1.37/2.11	2.78/2.79/4.54	**0.10/0.10/0.16**
CE-RP	TD	6.30/6.21/12.43	24.50/24.14/46.26	3.48/3.43/7.02	11.04/10.88/22.20	13.22/13.02/27.16	**2.75/2.72/5.71**
Wave-Gauss	TD	16.14/15.75/26.83	16.83/16.63/22.04	4.80/4.70/7.28	12.44/12.21/17.54	14.88/14.63/19.73	**4.43/4.32/7.10**
ACE	TD	1.30/1.45/2.33	3.01/3.18/3.24	1.20/2.32/3.94	0.10/0.11/0.20	0.13/0.15/0.31	**0.02/0.02/0.03**
Kolm-Smooth	TD	2.23/2.21/4.63	11.52/11.49/28.48	2.39/2.38/5.19	2.44/2.43/5.59	8.15/8.12/17.65	**1.16/1.16/2.47**
Kolm-Turb	TD	7.86/7.73/23.29	70.40/70.09/115.16	4.13/4.08/13.58	5.22/5.16/17.15	23.09/22.94/61.80	**2.00/1.96/6.98**
KS-Strong	TD	3.93/3.92/12.39	24.98/24.90/67.08	4.09/4.08/11.83	2.51/2.50/7.38	15.00/14.94/43.22	**0.52/0.52/1.66**
KS-Weak	TD	1.23/1.23/2.92	13.36/13.29/38.76	7.18/7.14/15.15	2.00/1.99/4.05	4.71/4.69/12.70	**0.23/0.23/0.53**

Table 2: Ablation study on ACE and FNS-KF. Each entry reports $\mathcal{E}_{\text{step}}/\mathcal{E}_{\text{full}}/\mathcal{E}_{\text{final}}$ in percent, lower is better. The best and second-best results for each metric within each dataset are shown in bold and underlined, respectively, including ties.

Variant	Params	ACE	FNS-KF
Spectral branch only	9M	0.021/0.023/0.033	0.115/0.115/0.188
Spatial branch only	16M	**0.012/0.014/0.026**	0.098/0.098/0.168
1-directional traversal	10M	0.020/0.021/0.031	0.106/0.106/0.177
8-directional traversal	13M	0.022/0.024/0.035	**0.096/0.096**/0.159
Full SSF-Net	11M	0.020/0.021/0.033	0.098/0.098/**0.157**

SSF-Net reduces $\mathcal{E}_{\text{step}}$, $\mathcal{E}_{\text{full}}$, and $\mathcal{E}_{\text{final}}$ by 54.4%, 54.5%, and 51.6% on average, respectively. Notably, these improvements are achieved with only 11M parameters, fewer than all compared baselines.

Performance across PDE families. SSF-Net exhibits particularly strong performance on fluid-dynamical, coupled transport, and reaction–diffusion problems. Relative to the corresponding second-best methods, its step-averaged error is reduced by 67.7% on NS-PwC, 73.9% on NS-Tracer-PwC, 64.3% on FNS-KF, and 80.0% on ACE. The gains are also pronounced on the two Kuramoto–Sivashinsky benchmarks: on KS-Strong, SSF-Net reduces the step-averaged, full-rollout, and final-step errors by 79.3%, 79.2%, and 77.5%, respectively, relative to scOT, the second-best method. On the Kolmogorov benchmarks, the corresponding reductions remain approximately 47–52%. Improvements are smaller but remain consistent on Wave-Gauss, where SSF-Net still achieves the lowest error under all three metrics. These results suggest that the proposed spatial–spectral fusion is effective across elliptic, parabolic, hyperbolic, fluid-dynamical, and coupled transport problems rather than being specialized to a single equation family.

Long-horizon rollout stability. Across the four 100-step APEBench benchmarks, SSF-Net reduces $\mathcal{E}_{\text{step}}$, $\mathcal{E}_{\text{full}}$, and $\mathcal{E}_{\text{final}}$ by 65.0%, 65.0%, and 63.7% on average, respectively, relative to the second-best result for each task. Figure 3 further examines how prediction errors accumulate during autoregressive rollout on Kolm-Turb. SSF-Net achieves the lowest test-averaged relative L_1 error at every forecast step, reaching 6.98% at step 100, compared with 13.58% for PDE-Transformer, the strongest baseline on this task. The error-increment curves show that SSF-Net generally accumulates error more slowly, with a final-step increment of approximately 0.26 percentage points, compared with 0.41 for PDE-Transformer. A smaller late-stage increment alone, however, does not imply accurate prediction: CNO's increments decline substantially after their early peak while its relative error exceeds 100%. Together, these results support improved empirical rollout stability for SSF-Net over the evaluated horizon, although its prediction error continues to grow.

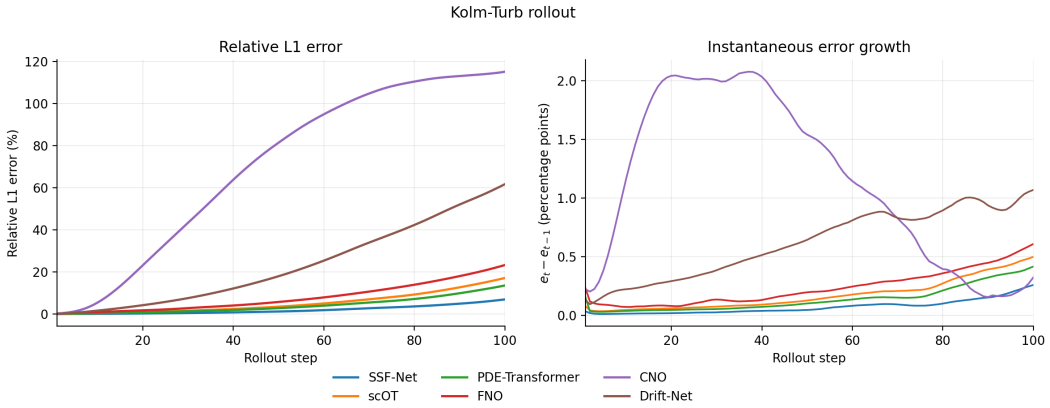

Figure 3: Error accumulation over 100 autoregressive rollout steps on Kolm-Turb. (a) relative L_1 error averaged over test trajectories at each step, reported in percent. (b) the corresponding step-to-step error increment $e_t - e_{t-1}$, reported in percentage points, with $e_0 = 0$.

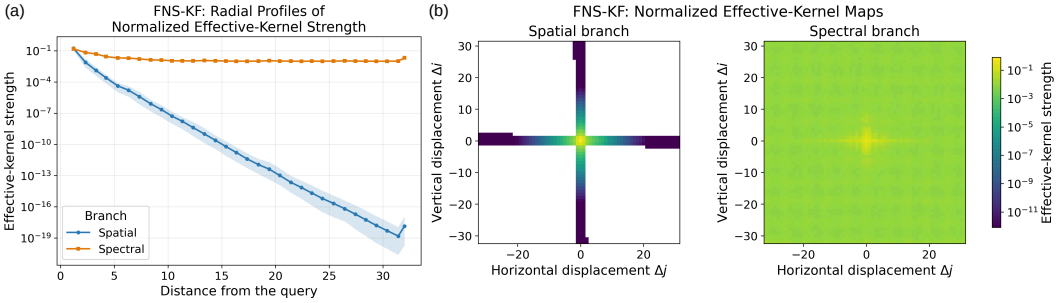

Figure 4: Effective-kernel analysis on FNS-KF at the first encoder block. (a) radially averaged normalized kernel strength, showing sample means and 2.5th–97.5th percentile bands. (b) sample-averaged normalized kernel strength of the spatial and spectral branches of SSF-Net. All panels share a logarithmic color scale, with gray indicating zero.

4.3 Ablation Study

We evaluate four variants of SSF-Net: (1) retaining only the spectral branch with doubled branch width. (2) retaining only the spatial branch with doubled branch width. (3) replacing the four-directional traversal with a single direction. and (4) extending the four axial traversals to eight by adding forward and reverse scans along both diagonal families (main-diagonal and anti-diagonal directions). Table 2 examines branch composition and traversal count on ACE and FNS-KF. ACE evolves through local nonlinear reaction and diffusion, making spatially adaptive processing a natural fit. Consistent with this structure, the wider spatial-only variant achieves the lowest errors on ACE, although it requires 16M parameters compared with 11M for the full model. FNS-KF combines advective transport with nonlocal pressure coupling under sustained forcing, motivating joint spatial and spectral processing. On this task, the full model achieves the lowest final-step error (0.157%), outperforming both single-branch variants. Four-directional traversal improves FNS-KF over a single direction but offers no gain on ACE. The competitive performance of a single direction on ACE may reflect its isotropic reaction–diffusion dynamics, for which additional scan directions provide limited benefit in these experiments. Eight directions yield only slightly lower average errors on FNS-KF at a higher parameter cost. These results suggest that the benefits of fusion and multi-directional traversal depend on the dynamics, with the full configuration offering a practical balance between accuracy and model size.

4.4 Model Analysis

To examine whether the two branches primarily focus on different interaction patterns—local dynamics in the spatial branch and global coupling in the spectral branch–we analyze the effective kernels of the first encoder block on FNS-KF using 16 test samples at the initial stored time, with the query fixed at the center of the feature grid. Effective-kernel strength measures how strongly input perturbations at each source location affect the query output, quantified by the Jacobian Frobenius norm across input and output channels. Each kernel is normalized by its full-map ℓ_2 norm separately for each sample and branch before sample averaging. Implementation details are provided in Appendix F.

As shown in Figure 4(b), the spatial branch concentrates its influence near the query in a cross-shaped pattern aligned with the four scan directions, whereas the spectral branch exhibits domain-wide responses with axis-aligned enhancements. Figure 4(a) radial profiles quantify this difference: spatial influence decays rapidly and approximately exponentially over most of the measured range, while spectral influence has a much flatter far-field profile. These complementary patterns agree with the analysis in Appendix C.

5 Limitations

SSF-Net has two main limitations. First, although its computational complexity is near-linear in the number of spatial tokens, its practical training and inference costs are higher than those of the compared baselines. Multi-directional scans, spectral transforms, and their implementation overhead limit the extent to which favorable asymptotic scaling translates into wall-clock efficiency.

Second, the spatial branch retains direction-dependent effective kernels. Unlike the spectral branch, which enables direct domain-wide interactions, spatial interactions are mediated by scan paths. Consequently, locations that are close in Euclidean distance may remain far apart along the traversal orders, causing their interactions to be strongly attenuated. Multi-directional traversal mitigates this mismatch but does not eliminate it, leaving some nearby interactions weakly represented by the spatial branch.

6 Conclusion

We introduced SSF-Net, a neural operator that combines multi-directional Mamba and Fourier-domain processing within a multiresolution encoder–decoder. The two branches provide complementary interaction patterns: spatial responses concentrate near the query and decay rapidly, while spectral responses extend across the domain. Across twelve PDE benchmarks, SSF-Net achieves lower prediction errors with fewer parameters than the compared baselines, and maintains its advantage over the evaluated 100-step autoregressive rollouts. Ablation studies reveal task-dependent trade-offs in branch composition and traversal count. Although SSF-Net retains near-linear asymptotic complexity, its practical computational overhead and direction-dependent spatial kernels remain limitations. Future work will focus on improving training and inference efficiency and developing spatial interaction mechanisms that better preserve Euclidean neighborhoods. SSF blocks can also replace the windowed attention blocks in Herde et al. (2024), and may be integrated into future foundation models.

AI use statement

We used generative AI tools to improve the grammar, clarity, and wording of the paper and to assist in checking the correctness of our mathematical derivations. The authors reviewed all AI-assisted suggestions and independently checked the mathematical arguments. We take full responsibility for the final content of the paper.

References

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Maximilian Herde, Bogdan Raonić, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel De Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for pdes. *Advances in Neural Information Processing Systems*, 37:72525–72624, 2024.

Benjamin Holzschuh, Georg Kohl, Florian Redinger, and Nils Thuerey. P3d: Scalable neural surrogates for high-resolution 3d physics simulations with global context. *arXiv preprint arXiv:2509.10186*, 2025a.

Benjamin Holzschuh, Qiang Liu, Georg Kohl, and Nils Thuerey. Pde-transformer: Efficient and versatile transformers for physics simulations. *arXiv preprint arXiv:2505.24717*, 2025b.

Felix Koehler, Simon Niedermayr, Rüdiger Westermann, and Nils Thuerey. Apebench: A benchmark for autoregressive neural emulators of pdes. *Advances in Neural Information Processing Systems*, 37:120252–120310, 2024.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Jiayi Li and Flora D. Salim. Drift-net: A spectral–coupled neural operator for pdes learning. *arXiv preprint arXiv:2509.24868*, 2025.

Zongyi Li. Neural operator: Learning maps between function spaces. In *2021 Fall Western Sectional Meeting*. AMS, 2021.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023a.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023b.

Shengjun Liu, Yu Yu, Ting Zhang, Hanchao Liu, Xinru Liu, and Deyu Meng. Architectures, variants, and performance of neural operators: A comparative review. *Neurocomputing*, 648:130518, 2025.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. *Advances in neural information processing systems*, 37:103031–103063, 2024.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries. *arXiv preprint arXiv:2502.02414*, 2025.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for physical surrogate models. *arXiv preprint arXiv:2310.02994*, 2023.

Alfio Quarteroni and Alberto Valli. *Numerical approximation of partial differential equations*. Springer, 1994.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel De Bézenac. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 36:77187–77200, 2023.

Louis Serrano, Thomas X Wang, Etienne Le Naour, Jean-Noël Vittaut, and Patrick Gallinari. Aroma: Preserving spatial structure for latent pde modeling with local neural fields. *Advances in Neural Information Processing Systems*, 37:13489–13521, 2024.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in neural information processing systems*, 35:1596–1611, 2022.

Yao Teng, Yue Wu, Han Shi, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Dim: Diffusion mamba for efficient high-resolution image synthesis. *arXiv preprint arXiv:2405.14224*, 2024.

Karn Tiwari, Niladri Dutta, NM Krishnan, et al. Latent mamba operator for partial differential equations. *arXiv preprint arXiv:2505.19105*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Sifan Wang, Jacob H Seidman, Shyam Sankaran, Hanwen Wang, George J Pappas, and Paris Perdikaris. Cvit: Continuous vision transformer for operator learning. *arXiv preprint arXiv:2405.13998*, 2024.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational visual media*, 8(3):415–424, 2022.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Jianwei Zheng, Wei Li, Ni Xu, Junwei Zhu, Xiaoxu Lin, and Xiaoqin Zhang. Alias-free mamba neural operator. *Advances in Neural Information Processing Systems*, 37:52962–52995, 2024.

A PROBLEM SETTING

We consider PDEs of the general time-dependent form

$$\begin{aligned} \partial_t u(x,t) + \mathcal{L}\left(u, \nabla_x u, \nabla_x^2 u, \dots\right) = 0, \quad \forall x \in D \subset \mathbb{R}^d, \quad t \in (0,T), \\ \mathcal{B}(u) = 0, \quad \forall (x,t) \in \partial D \times (0,T), \quad u(0,x) = a(x), \quad x \in D, \end{aligned} \tag{12}$$

where the solution is defined on a domain $\mathcal{X} \subset L^p(D;\mathbb{R}^n)$ for some $1 \le p < \infty$ and time horizon $T > 0$. Here $a \in \mathcal{X}$ is the initial condition, and $u \in C([0,T];\mathcal{X})$ is the corresponding solution. $\mathcal{L}$ and $\mathcal{B}$ denote the differential and boundary operators, respectively. The equation 12 also covers PDEs with high-order time derivatives, as well as time-independent PDEs with variable coefficients and source terms.

Time-independent PDEs can be recovered from equation 12 by taking the long-time limit $t \to \infty$, i.e., $\bar{u} = \lim_{t\to\infty} u$, yielding the time-independent counterpart

$$\mathcal{L}\left(\bar{u}(x), \nabla_x \bar{u}, \nabla_x^2 \bar{u}, \dots\right) = 0, \quad \forall x \in D, \qquad \mathcal{B}(\bar{u}) = 0, \quad \forall x \in \partial D. \tag{13}$$

The solution of equation 12 defines a map $\mathcal{S} : [0,T] \times \mathcal{X} \to \mathcal{X}$, such that for any $t \in [0,T]$ and initial condition $a \in \mathcal{X}$, the solution is $u(t) = \mathcal{S}(t,a)$. For time-independent PDEs, the corresponding solution map is $\bar{\mathcal{S}} : \mathcal{X} \to \mathcal{X}$, with $\bar{u} = \bar{\mathcal{S}}(a)$.

This formulation gives rise to the underlying operator learning task: given a data distribution $\mu \in \mathrm{Prob}(\mathcal{X})$ and an initial condition $a \sim \mu$, our goal is to find an approximation $\mathcal{S}^*$ that closely reproduces the true solution operator $\mathcal{S}$ in equation 12. Given only the initial condition (and boundary conditions), the learned operator should generate the full trajectory $\{\mathcal{S}^*(t,a)\}_{t\in[0,T]}$ that approximates the true trajectory $\{\mathcal{S}(t,a)\}_{t\in[0,T]}$.

While the formulation above is defined over continuous function spaces, in practice we approximate $\mathcal{S}$ using a discretized representation of the domain. We consider a two-dimensional spatial domain $D \subset \mathbb{R}^2$, discretized on a grid of size $H \times W$ with C channels. Our model learns a one-step operator $\mathcal{S}_\theta : u_t \mapsto u_{t+1}$ and is trained with the standard relative L^1 loss. At inference stage, given only the initial condition and boundary conditions, it generates the full trajectory $\{u_t\}_{t=1}^T$ through autoregressive rollout.

B SSM PRELIMINARIES

For completeness, we briefly recap the state-space formulation underlying Mamba (Gu & Dao, 2023). We refer readers to Gu et al. (2021) for the full derivation. A continuous-time SSM maps an input signal $x(t) \in \mathbb{R}^N$ to an output $y(t) \in \mathbb{R}^N$ through a hidden state $h(t) \in \mathbb{R}^L$. An SSM can be formulated as ordinary differential equations (ODEs) as follows:

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), \qquad y(t) = \mathbf{C}h(t), \tag{14}$$

where $\mathbf{A} \in \mathbb{R}^{L\times L}, \mathbf{B} \in \mathbb{R}^{L\times 1}, \mathbf{C} \in \mathbb{R}^{1\times L}$ are learnable parameters.

To integrate the SSMs into neural networks, discretization operation is necessary. We use zero-order hold (ZOH) as the discretization rule to discretize SSM with a timescale $\Delta \in \mathbb{R}$:

$$\begin{aligned} h[i] = \bar{\mathbf{A}}h[i-1] + \bar{\mathbf{B}}x[i], \qquad y[i] = \mathbf{C}h[i], \\ \bar{\mathbf{A}} = e^{\Delta\mathbf{A}}, \qquad \bar{\mathbf{B}} = (\Delta\mathbf{A})^{-1}(e^{\Delta\mathbf{A}} - \mathbf{I}) \cdot \Delta\mathbf{B}, \end{aligned} \tag{15}$$

$$y = x * \bar{\mathbf{K}}, \qquad \bar{\mathbf{K}} = (\mathbf{C}\bar{\mathbf{B}}, \mathbf{C}\bar{\mathbf{A}}\bar{\mathbf{B}}, ..., \mathbf{C}\bar{\mathbf{A}}^k\bar{\mathbf{B}}, ...), \tag{16}$$

where $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are the discrete counterparts of $\mathbf{A}$ and $\mathbf{B}$. Equation 16 is the convolution form realization of SSMs.

Mamba further makes $\mathbf{B}$, $\mathbf{C}$, and Δ input-dependent, allowing the model to selectively propagate or forget information along the sequence. With these input-dependent parameters, the equation 16 can be written as follows:

$$\begin{aligned} \mathbf{B}_t = \mathbf{B}(x_t), \qquad \mathbf{C}_t = \mathbf{C}(x_t), \qquad \Delta_t = \Delta(x_t), \\ \bar{\mathbf{A}}_t = e^{\Delta_t\mathbf{A}}, \qquad \bar{\mathbf{B}}_t = (\Delta_t\mathbf{A})^{-1}(e^{\Delta_t\mathbf{A}} - \mathbf{I})\Delta_t\mathbf{B}_t, \end{aligned} \tag{17}$$

$$y_t = \sum_{j=0}^{t} \mathbf{K}_{t,j} x_j, \qquad \mathbf{K}_{t,j} = \mathbf{C}_t(\prod_{i=j+1}^{t} \bar{\mathbf{A}}_i)\bar{\mathbf{B}}_j. \tag{18}$$

The matrix of Mamba is a diagonal matrix, $\mathbf{A} = \mathrm{Diag}(\zeta_1, ..., \zeta_n)$ with $\zeta_i \in \mathbb{R}, \forall i$. Unlike conventional linear time invariance (LTI) SSMs, where the kernel K depends only on the time difference $t-j$, selective SSMs produce an input-dependent kernel whose coefficients vary with both the input sequence and temporal positions. This breaks time-invariance and lets the model decide, per position, how much of the incoming signal to write into the state ($\mathbf{B}$), what to read out of it ($\mathbf{C}$), and how quickly to forget what it already holds (Δ). For PDE fields this is the mechanism by which the branch adapts its effective interaction range to the local state of the solution.

C AN OPERATOR-LEARNING VIEW OF SSF

SSF-Net is not discretization-invariant in the sense of Kovachki et al. (2023). It sits with other patch-based surrogates (Herde et al., 2024; Li & Salim, 2025; Holzschuh et al., 2025b). We nonetheless adopt equation 19 to characterize the kernels each branch can represent at a fixed resolution, not to establish a continuous limit. Both branches of the SSF block can be understood as parameterizations of a single kernel integral operator and differ only in the structure they impose on κ.

$$\mathcal{K}(X)(s) = \int_{\Omega} \kappa(s,t)\, X(t)\, \mathrm{d}t, \qquad s \in \Omega, \tag{19}$$

where $\Omega \subset \mathbb{R}^d$ is the spatial domain on which the discretized field is defined, s and t denote a query and a source location, and $\kappa : \Omega \times \Omega \to \mathbb{R}^{D\times D}$ is a matrix-valued kernel specifying how strongly the feature at t influences that at s. This view makes explicit why the two branches are complementary rather than redundant.

C.1 SPATIAL BRANCH

The spatial branch introduces a scan-distance-dependent attenuation. A discrete selective SSM layer can be interpreted as a Monte-Carlo approximation of the kernel integral equation 19, with the kernel parameterized by the state-space dynamics (Tiwari et al., 2025):

$$\begin{aligned} \kappa_{\mathrm{local}}(s,t) &= C(s)\Big(\prod_{i=t+1}^{s} \bar{A}(i)\Big)\bar{B}(t), \\ \bar{A}(i) &= \mathrm{diag}(\zeta_{i,1}, \zeta_{i,2}, \ldots, \zeta_{i,d}), \end{aligned} \tag{20}$$

where $\bar{A}(i)$ is diagonal, so the product reduces to an elementwise accumulation along the ordering. Two properties follow. First, the accumulated product acts as an input-dependent relative positional encoding: since $|\zeta_{i,\ell}| \le 1$ for every eigenvalue in a stable SSM, the transition product has a non-increasing magnitude as additional scan steps are accumulated. This introduces a scan-distance-dependent attenuation of κ_{local}. Second, $\bar{B}$, C and the $\bar{A}$ are all functions of the input field according to Appendix B, so $\kappa_{\mathrm{local}}(s,t)$ depends on the two locations separately rather than on their displacement. Consequently, the kernel need not depend only on displacement: both the interaction strength and its rate of attenuation can adapt to the local state of the solution.

The multi-directional traverse symmetrises the local kernel. Equation 20 describes a single traversal, and D.4 illustrates the multi-directional traverse mechanism. Because the SSM recurrence is causal in the ordering π_n, the kernel of direction n vanishes on half of the domain,

$$\kappa^{(n)}_{\mathrm{local}}(s,t) = 0, \quad \text{whenever} \quad \pi_n(t) > \pi_n(s), \tag{21}$$

writing $\rho = \max_{i,l} |\zeta_{i,l}| \le 1$ for the largest eigenvalue magnitude and assuming $\|C(s)\| \le 1$, $\|\bar{B}(t)\| \le 1$, its remaining entries obey

$$\big\|\kappa^{(n)}_{\mathrm{local}}(s,t)\big\| \lesssim \rho^{\,d_n(s,t)}, \qquad d_n(s,t) = \big|\pi_n(s) - \pi_n(t)\big|. \tag{22}$$

A single scan is therefore both one-sided and anisotropic. Writing the 2D coordinates $s = (i,j)$, $t = (i',j')$ and $(\Delta_i, \Delta_j) = (i - i',\, j - j')$, the raster orderings of equation 48 give

$$d_1 = d_2 = \big|\Delta_i\, W + \Delta_j\big|, \qquad d_3 = d_4 = \big|\Delta_j\, H + \Delta_i\big|, \tag{23}$$

where d_n, $n = 1, 2, 3, 4$ is the distance between s and t in different traverse orders. So under π_1 alone a horizontal neighbor sits at ordering distance 1 while a vertical neighbor sits at distance d_1 and is attenuated by ρ^{d_1} rather than ρ, a result of the flattening, not of the underlying PDE field.

Summing the four directions in equation 6 removes both one-sidedness and mitigates directional bias. Since $\pi_2 = N + 1 - \pi_1$ and $\pi_4 = N + 1 - \pi_3$, the four orderings form two reversed pairs, and for any ordered pair (s, t) exactly one member of each pair satisfies the causality condition above. The combined scan orderings therefore permit a dependency path between every pair of locations. However, this structural connectivity does not guarantee a nonzero kernel value, since learned weights may vanish or contributions from different directions may cancel. The magnitude is governed by the shortest of the four orderings,

$$\big\|\kappa_{\mathrm{local}}(s,t)\big\| \lesssim \rho^{\,d_\star(\Delta_i,\Delta_j)}, \qquad d_\star = \min\big(|\Delta_i W + \Delta_j|, |\Delta_j H + \Delta_i|\big). \tag{24}$$

The distance $d_\star$ is invariant under simultaneous sign reversal, $(\Delta_i, \Delta_j) \mapsto (-\Delta_i, -\Delta_j)$. For square grids ($H = W$), it is also invariant under exchanging the coordinate offsets, $(\Delta_i, \Delta_j) \mapsto (\Delta_j, \Delta_i)$. In particular $d_\star = 1$ for all four axis-aligned nearest neighbors, whereas any single ordering places two of them at distance W or H. The decay profile of the spatial branch is therefore balanced across both axes and both signs. We refer to this property as local isotropy. The property is exact on the coordinate axes but not for arbitrary displacements. For example, the offset $(1, 1)$ has $d_\star = \min(W + 1, H + 1)$, whereas its Euclidean distance is $\sqrt{2}$ in grid units, so purely diagonal interactions remain attenuated in the spatial branch and are instead carried by the spectral branch and by the multi-resolution hierarchy.

C.2 Spectral branch

The spectral branch permits global interactions without recurrent distance decay. Starting from the integral-operator representation in equation 19, we characterize the nonlinear spectral branch through its effective kernel:

$$\delta\mathcal{K}_{\mathrm{global}}(X)(s) = \int_\Omega \kappa_{\mathrm{global}}(s,t)\,\delta X(t)\,\mathrm{d}t, \qquad \kappa_{\mathrm{global}}(s,t) = \frac{\delta\mathcal{K}_{\mathrm{global}}(X)(s)}{\delta X(t)}. \tag{25}$$

As specified in equation 51, the nonlinearities in equation 9 and equation 10 act separately on the real and imaginary components. For a complex vector $z = x + iy$, consider the componentwise map $\phi_{\mathbb{C}}(z) = \phi(x) + i\phi(y)$. Define

$$D_r = \mathrm{diag}\big(\phi'(x)\big), \qquad D_i = \mathrm{diag}\big(\phi'(y)\big), \tag{26}$$

where ϕ' is evaluated elementwise. Its differential is

$$\delta\phi_{\mathbb{C}} = D_r\,\delta x + iD_i\,\delta y = \frac{D_r + D_i}{2}\,\delta z + \frac{D_r - D_i}{2}\,\overline{\delta z}. \tag{27}$$

The conjugate term is present whenever $D_r \neq D_i$, that is, when the derivatives evaluated at the real and imaginary components differ.

Suppressing the block index ℓ in equation 9, let P_k and Q_k denote the $\frac{D_r(k)+D_i(k)}{2}$ and $\frac{D_r(k)-D_i(k)}{2}$ of the real and imaginary ReLU derivative matrices, evaluated at $W_1 Z(k) + b_1$ in equation 27. Similarly, let R_k and S_k denote the $\frac{D_r(k)+D_i(k)}{2}$ and $\frac{D_r(k)-D_i(k)}{2}$ of the soft-shrinkage derivative matrices, evaluated at $\tilde{Z}(k)$. The derivatives are $\mathrm{ReLU}'(u) = \mathrm{diag}[u > 0]$ and $S'_\lambda(u) = \mathrm{diag}[|u| > \lambda]$ away from their thresholds. Applying equation 27 to equation 9 gives

$$\delta\tilde{Z}(k) = W_2 P_k W_1\,\delta Z(k) + W_2 Q_k \overline{W_1}\,\overline{\delta Z(k)}. \tag{28}$$

Differentiating equation 10 then yields

$$\begin{aligned}\delta\hat{Z}(k) &= R_k\,\delta\tilde{Z}(k) + S_k\,\overline{\delta\tilde{Z}(k)} \\ &= A_k\,\delta Z(k) + B_k\,\overline{\delta Z(k)},\end{aligned} \tag{29}$$

where

$$\begin{aligned}A_k &= R_k W_2 P_k W_1 + S_k \overline{W_2} Q_k W_1, \\ B_k &= R_k W_2 Q_k \overline{W_1} + S_k \overline{W_2} P_k \overline{W_1}.\end{aligned} \tag{30}$$

The coefficients depend on the input and frequency through the activation masks, although the learned weights are shared across frequencies. The biases b_1, b_2 affect these masks and are therefore retained in their evaluation. The formulas hold away from activation thresholds. At the thresholds, the derivatives are understood according to the automatic differentiation convention used by the implementation.

To recover the spatial kernel, we start from the equation 19 on the discrete grid $\hat{\Omega} = \mathbb{Z}_H \times \mathbb{Z}_W$:

$$\mathcal{K}(X)(s) = \sum_{t \in \hat{\Omega}} \kappa(s,t)\, X(t), \qquad s \in \hat{\Omega}. \tag{31}$$

We write $k \in \tilde{\Omega}, \tilde{\Omega} = \mathbb{Z}_H \times \mathbb{Z}_W$ for a frequency index and $\langle k, s \rangle = \frac{k_1 s_1}{H} + \frac{k_2 s_2}{W}$, so that a field X and its discrete Fourier transform $Z = \mathcal{F}(X)$ are related by

$$Z(k) = \frac{1}{\sqrt{N}} \sum_{s \in \hat{\Omega}} X(s)\, e^{-2\pi i \langle k, s \rangle}, \qquad X(s) = \frac{1}{\sqrt{N}} \sum_{k \in \tilde{\Omega}} Z(k)\, e^{2\pi i \langle k, s \rangle}. \tag{32}$$

For the stored half-spectrum $\Omega_+ = \mathbb{Z}_H \times \{0, \dots, W_r - 1\}$, let $\eta(k_2) = 1$ on the columns $k_2 = 0$ and, when W is even, $k_2 = W/2$, and let $\eta(k_2) = 2$ otherwise. These weights account for conjugate completion in the inverse real FFT.

Substituting equation 29 and equation 32 into the differential of equation 11, and exchanging the spatial and frequency sums, gives

$$\begin{aligned} \delta X_{\mathrm{global}}(s) = & \delta X_f(s) \\ & + \sum_{t \in \hat{\Omega}} \left[\frac{1}{N} \operatorname{Re} \sum_{k \in \Omega_+} \eta(k_2) \left(A_k e^{2\pi i \langle k, s-t \rangle} + B_k e^{2\pi i \langle k, s+t \rangle} \right) \right] \delta X_f(t). \end{aligned} \tag{33}$$

Consequently,

$$\begin{aligned} \kappa_{\mathrm{global}}(s,t) &= \delta_{s,t} I + a_f(s-t) + b_f(s+t) \\ &= \delta_{s,t} I + a_f(-\Delta) + b_f(2s + \Delta), \end{aligned} \tag{34}$$

where

$$\begin{aligned} a_f(u) &= \frac{1}{N} \operatorname{Re} \sum_{k \in \Omega_+} \eta(k_2) A_k e^{2\pi i \langle k, u \rangle}, \\ b_f(u) &= \frac{1}{N} \operatorname{Re} \sum_{k \in \Omega_+} \eta(k_2) B_k e^{2\pi i \langle k, u \rangle}, \\ \Delta &= t - s. \end{aligned} \tag{35}$$

The identity term comes from the residual in equation 11. The direct term depends on $s - t$ and is a convolution, whereas the conjugate term depends on $s + t$ and is a reflected convolution: it convolves b_f with the reflected perturbation $\delta X_f(-t)$. Thus, the effective kernel need not depend only on displacement.

Two properties follow. First, the Fourier sums in equation 35 permit support over the entire domain, allowing distant locations to interact without propagation through intermediate spatial positions. Second, unlike the transition products of the spatial branch, these sums impose no explicit monotonic or exponential distance-decay constraint. The absence of such an architectural constraint does not guarantee constant or nonzero kernel magnitude at large distances. The learned interactions may be oscillatory and input-dependent, with the reflected term additionally introducing dependence on the query and source locations separately.

C.3 SSF Block

The SSF block combines both kernel families. The two preceding analyses use different kernel representations to highlight the properties of each branch: the spatial analysis uses an input-dependent recurrence kernel for the SSM, whereas the spectral analysis uses a differential kernel to characterize perturbation propagation. For the forward-kernel interpretation below, we use

$\hat{\kappa}_{\mathrm{local}}$ and $\hat{\kappa}_{\mathrm{global}}$ to denote input-dependent coefficient kernels of the complete spatial and spectral branches, respectively. In particular, $\hat{\kappa}_{\mathrm{local}}$ incorporates the operations surrounding the SSM recurrence, including convolution, normalization, gating, and projections, while $\hat{\kappa}_{\mathrm{global}}$ is distinct from its corresponding differential kernel κ_{global} derived above. The additive offsets are omitted from the expressions below. Here, $\mathcal{K}(X)$ denotes the sum of the branch outputs before the pointwise fusion layers.

Splitting the channels assigns the two parameterizations to complementary subspaces, and summing the two branch outputs superposes them on a shared set of channels. Writing the block input as $X = [X_s; X_f]$ with $X_s, X_f \in \mathbb{R}^D$ and $D = C/2$, the kernel acting on the full channel space is the block row

$$\kappa_{\mathrm{SSF}}(s,t) = \begin{bmatrix} \hat{\kappa}_{\mathrm{local}}(s,t) & \hat{\kappa}_{\mathrm{global}}(s,t) \end{bmatrix}, \tag{36}$$

$$\begin{aligned} \mathcal{K}(X)(s) &= \int_\Omega \begin{bmatrix} \hat{\kappa}_{\mathrm{local}}(s,t) & \hat{\kappa}_{\mathrm{global}}(s,t) \end{bmatrix} \begin{bmatrix} X_s(t) \\ X_f(t) \end{bmatrix} \mathrm{d}t \\ &= \int_\Omega \hat{\kappa}_{\mathrm{local}}(s,t)\, X_s(t)\, \mathrm{d}t + \int_\Omega \hat{\kappa}_{\mathrm{global}}(s,t)\, X_f(t)\, \mathrm{d}t, \end{aligned} \tag{37}$$

so each of the D output channels draws on both kernel families. The summation fuses the two parameterizations. And the MLP in equation 4 then acts pointwise, composing with equation 36 as a pointwise nonlinear map that lifts the result back to C channels. The forward kernel $\hat{\kappa}_{\mathrm{local}}$ incorporates the recurrence kernel κ_{local} and the surrounding spatial-branch operations, while κ_{global} describes the differential of the spectral map represented by $\hat{\kappa}_{\mathrm{global}}$. These connections motivate complementary biases toward scan-mediated local interactions and direct global mixing. The spatial pathway supports input-adaptive interactions with attenuation along scan paths, whereas the spectral pathway permits direct domain-wide interactions without a recurrent distance-decay constraint. As shown in equation 34, the nonlinear spectral kernel may depend on both displacement and the query and source locations separately. It is not restricted to a translation-invariant convolution. These properties motivate explicit fusion of local and global interactions, without assuming that either branch exclusively represents one type of feature. The block combines the two pathways at a cost that remains near-linear in the sequence length.

D IMPLEMENTATION DETAILS

As shown in Figure 2, the spatial-spectral fusion (SSF) is constructed as an encoder-decoder structure. The original input is patched, processed by multiple stages of SSF blocks, and downsampled to coarser resolution. Skip connections link each encoder level to its corresponding decoder level through ConvNeXt blocks (Liu et al., 2022), except at the bottleneck, where no skip connection is used. In the decoders, the data is assembled through upsampling and patch recovery. In the following section, we will introduce the implementation details of our model.

D.1 OVERLAP PATCH

Patch partitioning creates the visual tokens. We extract overlapping patches with a sliding window of kernel size $q = p + o$ and stride p, where o denotes the overlap in pixels. Given an input $X_{\mathrm{in}} \in \mathbb{R}^{H\times W\times C}$, zero-padding is applied so that the window count is $H^{(1)} \times W^{(1)}$ with $H^{(1)} = \lceil H/p \rceil$ and $W^{(1)} = \lceil W/p \rceil$. Let (r_j, s_j) be the 2D index of the j-th window, $j = 1, \ldots, H^{(1)}W^{(1)}$. The corresponding window $y_j \in \mathbb{R}^{q\times q\times C}$ is

$$y_j(u,v,c') = X_{\mathrm{in}}\big((r_j-1)p+u,\ (s_j-1)p+v,\ c'\big), \qquad 1 \le u,v \le q. \tag{38}$$

Each patch is linearly embedded by a shared weight $V \in \mathbb{R}^{C^{(1)}\times C\times q\times q}$ and bias $b \in \mathbb{R}^{C^{(1)}}$:

$$v_j^{(c)} = \sum_{u=1}^{q}\sum_{v=1}^{q}\sum_{c'=1}^{C} V^{(c,c',u,v)}\, y_j(u,v,c') + b^{(c)}, \tag{39}$$

where c indexes the $C^{(1)}$ embedding channels. The embedding is then passed through a layer norm getting the result $h^{(1)} \in \mathbb{R}^{H^{(1)}\times W^{(1)}\times C^{(1)}}$.

D.2 SSF-NET

SSF Block At each resolution level in the U-shaped architecture, multiple SSF blocks are stacked. We index resolution levels by $\ell \in \{1, \dots, L_{\text{res}}\}$, where $\ell = 1$ is the finest level and $\ell = L_{\text{res}}$ the bottleneck. L_{res} denotes the number of levels.

Each SSF block wraps the split–process–fuse map of equation 2-4 in a residual connection. The details of the two branches are introduced in Appendix D.4 and Appendix D.5. At resolution level ℓ, the input $h^{(\ell)} \in \mathbb{R}^{H^{(\ell)} \times W^{(\ell)} \times C^{(\ell)}}$ is split into $h_s^{(\ell)} \in \mathbb{R}^{H^{(\ell)} \times W^{(\ell)} \times D^{(\ell)}}$ and $h_f^{(\ell)} \in \mathbb{R}^{H^{(\ell)} \times W^{(\ell)} \times D^{(\ell)}}$, $D^{(\ell)} = C^{(\ell)}/2$. $h_{\text{sum}}^{(\ell)} \in \mathbb{R}^{H^{(\ell)} \times W^{(\ell)} \times D^{(\ell)}}$ is the layer-normalized sum of the spatial and spectral branch outputs, $f_{\text{spatial}}(h_s^{(\ell)})$ and $f_{\text{spectral}}(h_f^{(\ell)})$, respectively. The fusion MLP of equation 4 is a two-layer pointwise network with GeLU activation and four times the width of the latent dimension, applied identically at every spatial location:

$$\begin{aligned} \text{MLP}(h_{\text{sum}}^{(\ell)}) &= \big(W_2\big(\sigma(W_1 h_{\text{sum}}^{(\ell)} + b_1)\big) + b_2\big), \\ W_1 \in \mathbb{R}^{D_h^{(\ell)} \times D^{(\ell)}}, &\qquad W_2 \in \mathbb{R}^{C^{(\ell)} \times D_h^{(\ell)}}, \end{aligned} \tag{40}$$

where σ is GeLU and $D_h^{(\ell)} = 4 \cdot D^{(\ell)}$ the hidden width. It carries no spatial mixing, so all coupling between locations is produced by the two branches. Its role is to recombine the fused channels and to lift the branch width $D^{(\ell)}$ back to the block width $C^{(\ell)}$.

SSF Stage Let $\mathcal{M}_i^{(\ell,*)}$ denote i-th SSF block with residual at level ℓ, where $* \in \{\text{enc}, \text{dec}\}$ indicates whether the block belongs to the encoder or the decoder, and write $\text{SSF}_i^{(\ell,*)}$ for the split–process–fuse map of equation 2-equation 4 instantiated at level ℓ. Each block applies it residually,

$$\mathcal{M}_i^{(\ell,*)}(h_i^{(\ell)}) = h_i^{(\ell)} + \text{SSF}_i^{(\ell,*)}(h_i^{(\ell)}). \tag{41}$$

where $h_i^{(\ell)} \in \mathbb{R}^{H^{(\ell)} \times W^{(\ell)} \times C^{(\ell)}}$ denotes the input for i-th SSF block in ℓ-th SSF Stage. So a stage is a composition of residual blocks. The stage at level ℓ is

$$\mathcal{S}^{(\ell,*)} = \mathcal{M}_{n^{(\ell)}}^{(\ell,*)} \circ \mathcal{M}_{n^{(\ell)}-1}^{(\ell,*)} \circ \cdots \circ \mathcal{M}_1^{(\ell,*)}, \qquad \ell = 1, \dots, L_{\text{res}}, \quad * \in \{\text{enc}, \text{dec}\}, \tag{42}$$

where $n^{(\ell)}$ is the number of SSF blocks at level ℓ. The encoder and decoder use the same depth n_ℓ at a given level, but $\mathcal{S}^{(\ell,\text{enc})}$ and $\mathcal{S}^{(\ell,\text{dec})}$ have independent parameters, as do stages at different levels.

Encoder, bottleneck and decoder. Write $\mathcal{D}^{(\ell)}$ for the downsampling operator from level ℓ to $\ell+1$, a convolution with kernel size 3, stride 2 and padding 1 that halves both spatial dimensions and doubles the channel width. $\mathcal{U}^{(\ell)}$ for the corresponding upsampling operator, a transposed convolution with kernel size 2 and stride 2 that reverses both, and $\mathcal{G}^{(\ell)}$ for the skip path at level ℓ, a stack of $n_{\text{skip}}^{(\ell)}$ ConvNeXt blocks in their original form (Liu et al., 2022) that leaves the width $C^{(\ell)}$ unchanged. With $h^{(1)}$ the output of the overlapping patch embedding (equation 1), the encoder proceeds as

$$e^{(\ell)} = \mathcal{S}^{(\ell,\text{enc})}(h^{(\ell)}), \qquad h^{(\ell+1)} = \mathcal{D}^{(\ell)}(e^{(\ell)}), \qquad \ell = 1, \dots, L_{\text{res}} - 1. \tag{43}$$

The bottleneck is at the coarsest level and carries no skip connection,

$$d^{(L_{\text{res}})} = \mathcal{S}^{(L_{\text{res}},\text{dec})} \mathcal{S}^{(L_{\text{res}},\text{enc})}(h^{(L_{\text{res}})}), \tag{44}$$

and the decoder ascends symmetrically, fusing the skip by addition before the stage,

$$d^{(\ell)} = \mathcal{S}^{(\ell,\text{dec})}\big(\mathcal{U}^{(\ell)}(d^{(\ell+1)}) + \mathcal{G}^{(\ell)}(e^{(\ell)})\big), \qquad \ell = L_{\text{res}} - 1, \dots, 1. \tag{45}$$

The prediction is obtained by applying the de-patching operator of Appendix D.3 to $\mathbf{d}^{(1)}$. The total number of SSF blocks is $2 \sum_{\ell=1}^{L_{\text{res}}} n^{(\ell)}$.

D.3 DE-PATCHING

De-patching inverts the operation of Appendix D.1, mapping the finest decoder output back to a field on the original $H \times W$ grid with C_{out} physical channels. Re-indexing the patch index $j \in$

$\{1, \ldots, H^{(1)} \cdot W^{(1)}\}$ of equation 39 as a grid pair (m, n), $m \in \{1, \ldots, H^{(1)}\}, n \in \{1, \ldots, W^{(1)}\}$, let $d^{(c)}(m, n)$, $c = 1, \ldots, C^{(1)}$, denote the decoder output at the finest resolution.

A transposed convolution with kernel size and stride both equal to p scatters each token onto a $p \times p$ block. Writing the pixel index as $i = (m-1)p + u$, $j = (n-1)p + v$ with $u, v \in \{1, \ldots, p\}$,

$$r^{(c)}(i, j) = \sum_{c'=1}^{C^{(1)}} V^{(c,c',u,v)}\, d(m, n, c') + b^{(c)}, \qquad V \in \mathbb{R}^{C^{(1)} \times C^{(1)} \times p \times p}, \quad b \in \mathbb{R}^{C^{(1)}}. \tag{46}$$

Unlike the embedding kernel of size $p+o$, this kernel has no overlap: each pixel is written by exactly one token, so $r \in \mathbb{R}^{H \times W \times C^{(1)}}$ is discontinuous at patch borders. A 3×3 convolution then projects back to the n physical channels while mixing across those borders,

$$\begin{aligned} u^{(c)}(i, j) = \sum_{\delta=-1}^{1} \sum_{\varepsilon=-1}^{1} \sum_{c'=1}^{C^{(1)}} V'^{(c,c',\delta,\varepsilon)}\, r(i+\delta, j+\varepsilon, c') + \hat{b}^{(c)}, \\ V' \in \mathbb{R}^{C_{\text{out}} \times C^{(1)} \times 3 \times 3}, \qquad \hat{b} \in \mathbb{R}^{C_{\text{out}}}. \end{aligned} \tag{47}$$

the out-of-range indices resolved by a padding rule matching the boundary condition of the task. The asymmetry with the embedding is deliberate: overlap on the encoder side aggregates neighbouring context into each token, whereas an overlapping transposed convolution would give each output pixel a spatially non-uniform number of contributions.

D.4 SPATIAL BRANCH

As shown in equation 2, each input $X \in \mathbb{R}^{H \times W \times C}$ is split into $X_s, X_f \in \mathbb{R}^{H \times W \times D}$, $D = C/2$ for the two branches. The multi-directional Mamba is used in spatial branch in the SSF blocks.

Multi-directional traverse We define the traverse operator π_n in equation 5 in this paragraph. Let $\Omega_{\text{pos}} = \{(i, j) : 1 \le i \le H,\ 1 \le j \le W\}$ denote the grid indices at the current resolution, with $N = HW$. Each traversal direction $n \in \{1, 2, 3, 4\}$ is a bijection $\pi_n : \Omega_{\text{pos}} \to \{1, \ldots, N\}$ assigning to every grid point its position in the flattened sequence:

$$\begin{aligned} \pi_1(i, j) &= (i-1)W + j, & \pi_2(i, j) &= N + 1 - \pi_1(i, j), \\ \pi_3(i, j) &= (j-1)H + i, & \pi_4(i, j) &= N + 1 - \pi_3(i, j), \end{aligned} \tag{48}$$

corresponding to row-major, reversed row-major, column-major, and reversed column-major orderings. Each ordering is a raster scan: every row (column) is read in the same direction. This keeps the ordering consistent with the reversal identities $\pi_2 = N + 1 - \pi_1$ and $\pi_4 = N + 1 - \pi_3$, so the four directions form two reversed pairs. The inverse maps π_n^{-1} are obtained by the same formulas read backwards and are applied to restore the grid layout.

Selective SSM. In a selective SSM, Δ, $\mathbf{B}$ and $\mathbf{C}$ are input-dependent. Each scan direction $n \in \{1, 2, 3, 4\}$ forms a group with its own state-space parameters $\mathbf{B}_n$ and $\mathbf{C}_n$, and each group is split into M parallel heads that share $\mathbf{B}_n$ and $\mathbf{C}_n$ but carry their own timescale Δ_n^m and decay $\mathbf{A}_n^m$:

$$\begin{aligned} \mathbf{A}_n^m &= -\exp(\alpha_n^m), \\ \Delta_n, \mathbf{B}_n, \mathbf{C}_n &= \text{Linear}_n(X_n), \\ \bar{\mathbf{A}}_n^m, \bar{\mathbf{B}}_n &= \text{discretize}\big(\Delta_n^m, \mathbf{A}_n^m, \mathbf{B}_n\big), \qquad m = 1, \ldots, M, \\ X_{\text{ssm}} = \sum_{n=1}^{4} \pi_n^{-1}\Big(\text{Concat}\big(&\text{SSM}(\bar{\mathbf{A}}_n^1, \bar{\mathbf{B}}_n, \mathbf{C}_n)(X_n^1), \ldots, \text{SSM}(\bar{\mathbf{A}}_n^M, \bar{\mathbf{B}}_n, \mathbf{C}_n)(X_n^M)\big)\Big). \end{aligned} \tag{49}$$

where α_n^m is a learnable parameter, $\text{SSM}(\bar{\mathbf{A}}_n^m, \bar{\mathbf{B}}_n, \mathbf{C}_n)(X_n^m)$ denotes the recurrence equation 18 run with $(\bar{\mathbf{A}}_n^m, \bar{\mathbf{B}}_n, \mathbf{C}_n)$ on the m-th channel slice of X_n, and π_n^{-1} is the inverse of the traversal π_n. Only Δ_n, $\mathbf{B}_n$ and $\mathbf{C}_n$ are produced from the input. $\mathbf{A}_n^m$ is a learned scalar. Within a direction, all heads share the same $(\mathbf{B}_n, \mathbf{C}_n)$ and differ only in $\mathbf{A}_n^m$.

D.5 Spectral Branch: Nonlinearity on the complex domain

Both the ReLU of equation 9 and the soft-shrinkage of equation 10 act in the spectral domain, where the features are complex-valued. Since complex field $\mathbb{C}$ admits no natural ordering, neither is defined on a complex value as a whole. We therefore identify $\mathbb{C}$ with $\mathbb{R}^2$ and apply both nonlinearities to the real and imaginary parts independently. Writing

$$\mathcal{R} : \mathbb{C} \to \mathbb{R}^2, \qquad \mathcal{R}(a + i\,b) = (a,\, b), \tag{50}$$

for this identification, a real-valued nonlinearity ϕ is lifted to the complex domain as

$$\phi_{\mathbb{C}}(z) = \mathcal{R}^{-1}\big(\phi(\mathcal{R}(z))\big), \qquad \text{i.e.} \qquad \phi_{\mathbb{C}}(a + i\,b) = \phi(a) + i\,\phi(b), \tag{51}$$

where ϕ acts elementwise. We use equation 51 for both $\phi = \mathrm{ReLU}$ and $\phi = S_\lambda$.

Complex linear layer. Writing a complex weight as $W = W_r + i\,W_i$ with W_r, W_i real, the product Wz for $z = x_r + i\,x_i$ expands to

$$\mathcal{R}(Wz) \;=\; \begin{pmatrix} W_r & -W_i \\ W_i & W_r \end{pmatrix} \begin{pmatrix} x_r \\ x_i \end{pmatrix}, \tag{52}$$

so a complex linear layer is realised as a real linear layer with the structure of equation 52, and both of its outputs are real-valued. Composing equation 52 with equation 51, one layer of the block-diagonal complex MLP in equation 9 is computed as

$$\mathcal{R}\big(\mathrm{ReLU}_{\mathbb{C}}(Wz)\big) = \mathrm{ReLU}\left(\begin{pmatrix} W_r & -W_i \\ W_i & W_r \end{pmatrix} \begin{pmatrix} x_r \\ x_i \end{pmatrix} \right). \tag{53}$$

Soft-shrinkage. The same treatment applies to the sparsification of equation 10. Writing $\tilde{z} = \tilde{z}_r + i\,\tilde{z}_i$ for an entry of $\tilde{Z}$, the threshold is applied to each part separately,

$$\hat{z} \;=\; S_\lambda(\tilde{z}_r) \;+\; i\,S_\lambda(\tilde{z}_i), \qquad S_\lambda(x) = \mathrm{sign}(x)\max\{|x| - \lambda,\, 0\}, \tag{54}$$

with λ a fixed threshold, shared across all levels and channels and not learned. An entry is zeroed if and only if the absolute values of both its real and imaginary parts are at most λ. For nonzero outputs, the operation reduces the magnitude and may also change the phase.

E Experiments and hyperparameters

E.1 Dataset Preprocessing and Input Construction

PDE GYM datasets As shown in Table 3, we use the public PDE GYM datasets introduced by Herde et al. (2024), including Poisson-Gauss, NS-PwC, NS-SL, NS-Tracer-PwC, FNS-KF, CE-RP, Wave-Gauss, and ACE. We use the original train/validation/test splits, spatial resolutions, temporal sampling intervals, and physical variables. Unless otherwise stated, preprocessing and normalization follow the official PDE GYM protocol. For time-dependent tasks, consecutive states (u_t, u_{t+1}) are used to construct one-step training pairs. For tasks with static conditioning fields, the normalized static field is concatenated with the dynamic state along the channel dimension and supplied to the model at every rollout step. In particular, Wave-Gauss uses the spatially varying wave-speed field, whereas FNS-KF uses the prescribed forcing field. The model predicts only the dynamic solution channels, while the static conditioning channels are preserved throughout autoregressive rollout.

APEBench-generated datasets We additionally generate four two-dimensional long-horizon datasets using APEBench (Koehler et al., 2024): Kolm-Turb, Kolm-Smooth, KS-Strong, and KS-Weak. Each dataset contains 4000 trajectories on a 128×128 grid, divided into 3,700 training, 200 validation, and 100 test trajectories. The first 3,900 trajectories are generated using the APEBench training-data generator, while the remaining 100 test trajectories are generated separately using its test-data generator. Each trajectory contains 101 states, corresponding to 100 state transitions, following 500 numerical warm-up steps.

Table 3: PDE GYM and APEBench-generated datasets used in our experiments. TI and TD denote time-independent and time-dependent tasks, respectively. All datasets are represented on 128×128 Cartesian grids.

Dataset	Governing equation	Type	Model input	Prediction target	Static conditioning
PDE GYM					
Poisson-Gauss	Poisson	TI	f	u	–
NS-PwC	Incompressible Navier–Stokes	TD	(u_x^t, u_y^t)	(u_x^{t+1}, u_y^{t+1})	–
NS-Tracer-PwC	Navier–Stokes with passive tracer	TD	$(u_x^t, u_y^t, c_{\text{tr}}^t)$	$(u_x^{t+1}, u_y^{t+1}, c_{\text{tr}}^{t+1})$	–
NS-SL	Incompressible Navier–Stokes	TD	(u_x^t, u_y^t)	(u_x^{t+1}, u_y^{t+1})	–
FNS-KF	Forced incompressible Navier–Stokes	TD	(u_x^t, u_y^t, g)	(u_x^{t+1}, u_y^{t+1})	g
CE-RP	Compressible Euler	TD	$(\rho^t, u_x^t, u_y^t, p^t)$	$(\rho^{t+1}, u_x^{t+1}, u_y^{t+1}, p^{t+1})$	–
Wave-Gauss	Acoustic wave equation	TD	(u^t, c_{w})	u^{t+1}	c_{w}
ACE	Allen–Cahn	TD	u^t	u^{t+1}	–
APEBench-generated					
Kolm-Smooth	Forced Navier–Stokes (vorticity form)	TD	u^t	u^{t+1}	–
Kolm-Turb	Forced Navier–Stokes (vorticity form)	TD	u^t	u^{t+1}	–
KS-Strong	Kuramoto–Sivashinsky	TD	u^t	u^{t+1}	–
KS-Weak	Kuramoto–Sivashinsky	TD	u^t	u^{t+1}	–

Notation. f and u denote the source and solution fields in the Poisson equation, respectively. (u_x, u_y) are the two velocity components. ρ and p denote fluid density and pressure. g is the prescribed forcing field supplied as a static input channel in FNS-KF. c_{tr} is the passive-tracer concentration, and c_{w} is the spatially varying wave speed. For Kolm-Smooth and Kolm-Turb, u denotes scalar vorticity. For KS-Strong and KS-Weak, it denotes the scalar state field. The prescribed forcing in the Kolmogorov-flow datasets is included in the data-generating dynamics but is not supplied as an additional model input. Superscripts t and $t+1$ indicate consecutive time steps.

KOLMOGOROV-FLOW DATASETS. We generate the Kolm-Turb and Kolm-Smooth datasets by solving the two-dimensional incompressible Navier–Stokes equation in vorticity form on a periodic domain:

$$\frac{\partial u}{\partial t} = -b \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} \odot \nabla(\Delta^{-1} u) \right) \cdot \nabla u + \nu \Delta u + \lambda u - k \cos\left(k \frac{2\pi}{L} y \right), \tag{55}$$

where $u = u(\mathbf{x}, t)$ denotes the scalar vorticity field. The coefficient b controls the strength of nonlinear vorticity advection, ν is the kinematic viscosity, and $\lambda < 0$ is a linear drag coefficient that removes energy accumulated in the low-frequency modes. The last term is the Kolmogorov forcing, where k is the injection wavenumber and L is the side length of the periodic spatial domain.

Both Kolm-Turb and Kolm-Smooth use the same forcing wavenumber $k = 4$, forcing scale 1.0, stored time interval $\Delta t = 0.1$, and 20 numerical substeps per stored transition. The two configurations differ only in the viscosity ν and linear drag coefficient λ. Kolm-Turb uses $\nu = 1.0 \times 10^{-2}$ and $\lambda = -0.1$ to produce a more turbulent and challenging long-horizon forecasting regime, whereas Kolm-Smooth uses $\nu = 1.5 \times 10^{-2}$ and $\lambda = -0.15$ exhibiting smoother spatial fields and less intense spatiotemporal variation.

KURAMOTO–SIVASHINSKY DATASETS. KS-Strong and KS-Weak are generated from the Kuramoto-Sivashinsky equation:

$$\frac{\partial u}{\partial t} = -\frac{b}{2} \|\nabla u\|_2^2 - \nu \nabla \cdot \nabla u - \zeta \, \mathbf{1}^\top \nabla^4 u, \tag{56}$$

where $u = u(\mathbf{x}, t)$ is the scalar state field. The coefficient b controls the nonlinear gradient-norm term. ν controls the destabilizing second-order term, and ζ controls the stabilizing fourth-order hyperdiffusion term. The competition between the second-order instability, fourth-order dissipation, and nonlinear gradient interaction gives rise to complex spatiotemporal dynamics.

Although Equation equation 56 is expressed using the physical coefficients b, ν, and ζ, our datasets are generated through the difficulty-based interface of APEBench. Both KS-Strong and KS-Weak use the same gradient-norm difficulty coefficient $\delta_2 = -6.0$ and fourth-order hyperdiffusion difficulty coefficient $\gamma_4 = -15.0$. The two configurations differ only in the second-order diffusion difficulty coefficient γ_2. KS-Strong uses $\gamma_2 = -1.2$, resulting in a stronger destabilizing anti-diffusion effect and producing more intense spatiotemporal variations, whereas KS-Weak uses $\gamma_2 = -0.8$, which weakens the second-order instability and produces comparatively milder and smoother dynamics.

E.2 Training and metrics

We train a one-step operator $\mathcal{S}_\theta : u_t \mapsto u_{t+1}$ with the standard relative L^1 loss. For Drift-Net, we use frequency-weighted loss proposed in Li & Salim (2025). At test stage, $\mathcal{S}_\theta$ runs autoregressively to generate the full trajectory $\{u_t\}_{t=1}^T$. We report step-averaged relative L_1 error $\mathcal{E}_{\text{step}}$, full-rollout relative L_1 error $\mathcal{E}_{\text{full}}$ and final-step relative L_1 error $\mathcal{E}_{\text{final}}$:

$$\mathcal{E}_{\text{step}} = \frac{1}{N_{\text{test}} T} \sum_{i=1}^{N_{\text{test}}} \sum_{t=1}^{T} \frac{\|\widehat{u}_i^t - u_i^t\|_1}{\|u_i^t\|_1}, \tag{57}$$

$$\mathcal{E}_{\text{full}} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\sum_{t=1}^{T} \|\widehat{u}_i^t - u_i^t\|_1}{\sum_{t=1}^{T} \|u_i^t\|_1}, \tag{58}$$

$$\mathcal{E}_{\text{final}} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \frac{\|\widehat{u}_i^T - u_i^T\|_1}{\|u_i^T\|_1}, \tag{59}$$

where u_i^t and $\widehat{u}_i^t$ denote the ground-truth field and the corresponding autoregressive prediction at forecast step t of the i-th test trajectory. N_{test} is the number of test trajectories, and T is the number of forecast steps. Each rollout starts from the ground-truth initial state $\widehat{u}_i^0 = u_i^0$.

The step-averaged error assigns equal importance to every forecast step. The full-rollout error evaluates the predicted trajectory as a whole and accounts for the relative contribution of states with different magnitudes. The final-step error focuses on the longest prediction horizon and is particularly sensitive to accumulated autoregressive errors and instability. Reporting all three metrics therefore provides a more complete assessment of average per-step accuracy, overall trajectory fidelity, and long-term rollout stability than any individual metric alone.

For tasks with multiple quantities of interest (QoIs), we follow the setting of Li & Salim (2025) computing the relative L_1 error for each QoI and take an unweighted average.

E.3 Training and Optimization

Training objective. For all time-dependent datasets, we train a one-step neural operator $\mathcal{S}_\theta : u_t \mapsto u_{t+1}$ using single-step teacher forcing. Each pair of consecutive ground-truth states (u_t, u_{t+1}) constitutes one supervised training example. Unless otherwise stated, models are optimized using the relative L_1 loss described in Section E. For multi-component datasets, the loss is computed separately for each predefined group of physical variables and then averaged across groups, preventing variables with larger channel counts from dominating the objective. Drift-Net instead uses its original frequency-weighted loss, following the configuration proposed in Li & Salim (2025). For the time-independent Poisson-Gauss task, the model is trained directly to predict the steady-state solution u from the source field f.

Optimization. We optimize all models using AdamW. Unless an architecture-specific configuration is stated otherwise, we use an initial learning rate of 5×10^{-4}, a weight decay of 10^{-6}, and a batch size of 40. Gradients are clipped to a maximum ℓ_2 norm of 5. For SSF-Net, FNO, PDE-Transformer, and scOT, the learning rate is linearly increased from 10^{-3} of its initial value to the full initial learning rate during the first two epochs, followed by cosine annealing to a minimum learning rate of 10^{-6}. CNO uses a batch size of 32 and a step-decay schedule that multiplies the learning rate by 0.9 every 10 epochs. Drift-Net uses its original initial learning rate of 3×10^{-4} together with the same linear-warmup and cosine-decay schedule.

Epoch budget and checkpoint selection. All methods are assigned the same maximum number of training epochs on each dataset. We train for 20 epochs on NS-SL, 100 epochs on Poisson-Gauss, and 40 epochs on all remaining benchmarks. At the end of each epoch, we evaluate the model on the validation split. For time-dependent tasks, checkpoint selection is based on the full-rollout relative L_1 validation error. For Poisson-Gauss, it is based on the relative L_1 error of the predicted steady-state solution. The test set is not used for optimization or checkpoint selection.

Hardware and precision. All experiments are implemented in PyTorch and conducted on a single NVIDIA H200 GPU in full single precision (FP32), without mixed-precision or distributed training.

E.4 Model Configurations

All models are trained from scratch separately on each dataset. For a given task, the input and output dimensions are adjusted according to the corresponding physical variables listed in Table 3, while the remaining architectural configuration is kept fixed across datasets.

SSF-Net. The input field is embedded using a convolutional projection with patch size 2 and base feature dimension 48. The model contains 4 resolution levels with 2 SSF blocks at each level, resulting in the depth configuration $(2, 2, 2, 2)$. The channel dimension is doubled after each encoder downsampling stage and symmetrically reduced in the decoder. Both downsampling and upsampling are implemented using learnable convolutional operators with kernel size 3, stride 2 and padding 1. Skip connections link the corresponding encoder and decoder levels, with $(2, 2, 2, 0)$ ConvNeXt blocks applied to the skip pathways from the shallowest to the deepest level, respectively. The resulting model contains approximately 11M trainable parameters.

FNO. Our FNO baseline follows the standard two-dimensional Fourier Neural Operator architecture (Li et al., 2020a). The input is lifted to a hidden width of 96 and processed by 5 Fourier layers. Each Fourier layer retains 20 modes along each spatial dimension and combines the spectral convolution with a pointwise linear transformation. The final features are projected to the task-specific output channels. This configuration contains approximately 36M trainable parameters.

CNO. We use the two-dimensional Convolutional Neural Operator (Raonic et al., 2023) with 4 encoder and decoder resolution levels. The channel multiplier is set to 32, with 8 residual blocks at each resolution level and 8 additional residual blocks in the bottleneck. The model uses 3×3 convolutional kernels and CNO-specific filtered activation functions. This configuration contains approximately 18M trainable parameters.

PDE-Transformer. Considering we train the model on a single dataset from scratch, we only use the PDE-Transformer (MC)(Holzschuh et al., 2025b) as the baseline. The PDE-Transformer baseline uses a 5-level multiscale encoder–decoder with patch size 4, base embedding dimension 96 and block depth $(2, 5, 8, 5, 2)$. The resulting model contains approximately 33M trainable parameters.

scOT. We use the scOT architecture introduced in Poseidon (Herde et al., 2024). The model employs a 4-level multiscale encoder–decoder with Swin-style window attention, patch size 4, base embedding dimension 48 and 4 blocks at each resolution level, giving the depth configuration $(4, 4, 4, 4)$. Skip connections link the corresponding encoder and decoder levels, with $(2, 2, 2, 0)$ ConvNeXt blocks. This configuration contains approximately 20M trainable parameters.

Drift-Net. Our Drift-Net baseline follows the spatial–spectral architecture of Li & Salim (2025). It uses a 4-level encoder–decoder with patch size 4, base embedding dimension 48 and block depths $(4, 4, 4, 4)$. Each block combines an image-space branch (ConvNeXt-style block with depthwise 3×3 convolution, pointwise 1×1 convolution) with a spectral branch for frequency-domain feature mixing. We retain the original frequency-weighted training objective of Drift-Net. The resulting configuration contains approximately 17M trainable parameters.

F Model Analysis

F.1 Effective-Kernel definition

Measurement setup. For the effective-kernele strength in Figure 1, Figure 4 and Figure 5, we use 16 samples at the initial stored time index and process each sample separately. A forward hook captures the input features of the first block in the first encoder stage. We split the captured features into the spatial and spectral channel groups and evaluate each branch independently. For other baselines in Figure 1, we only capture the feature in the first encoder stage without split. The query is the central feature location q.

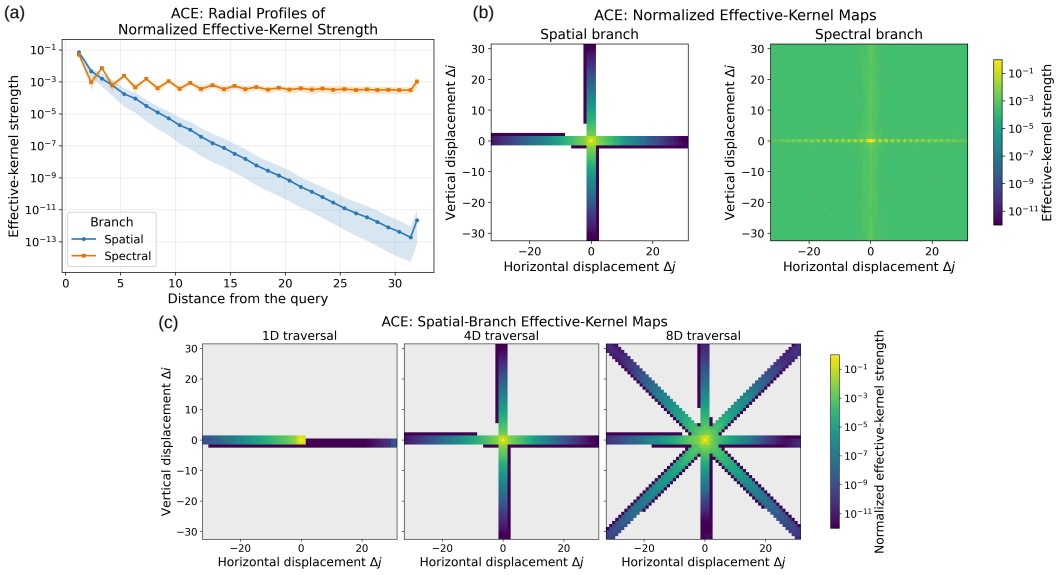

Figure 5: Effective-kernel analysis on ACE at the first encoder block. (a) radially averaged normalized kernel strength, showing sample means and 2.5th–97.5th percentile bands. (b) sample-averaged normalized kernel strength of the spatial and spectral branches of SSF-Net. (c) spatial-branch maps with one, four, and eight traversal directions. All panels share a logarithmic color scale, with gray indicating zero.

Table 4: Decay rate α

Branch	ACE	FNS-KF
Spatial branch	0.984	1.522
Spectral branch	0.070	0.044

Jacobian-based kernel strength. Let G_b denote the implemented branch map and $X_b^{(n)}$ its captured input for sample n, where $b \in \{\text{spatial}, \text{spectral}\}$. For a source location x, we compute

$$J_b^{(n)}(q,x) = \left.\frac{\partial G_b(X_b)(q)}{\partial X_b(x)}\right|_{X_b = X_b^{(n)}}, \qquad K_b^{(n)}(q,x) = \left\| J_b^{(n)}(q,x) \right\|_F. \tag{60}$$

The Frobenius norm sums squared derivatives over all input and output channels. We compute it by reverse-mode automatic differentiation, enumerating every output-channel basis vector at q and summing the squared input gradients. Each sample and branch is normalized separately using the entire feature grid Ω_h:

$$\widetilde{K}_b^{(n)}(q,x) = \frac{K_b^{(n)}(q,x)}{\left(\sum_{x' \in \Omega_h} K_b^{(n)}(q,x')^2\right)^{1/2}}, \qquad M_b(\delta) = \frac{1}{N}\sum_{n=1}^{N} \widetilde{K}_b^{(n)}(q, q+\delta), \tag{61}$$

where $N = 16$ and δ denotes displacement from the query.

Radial profiles. We group valid source displacements into unit-width annuli,

$$A_k = \{\delta : \; q+\delta \in \Omega_h, \; 1 \le \|\delta\|_2 \le 32, \; k \le \|\delta\|_2 < k+1\},$$
$$d_b^{(n)}(k) = \frac{1}{|A_k|} \sum_{\delta \in A_k} \widetilde{K}_b^{(n)}(q, q+\delta). \tag{62}$$

Only positions inside the feature grid are included. $d_b^{(n)}(k)$ denotes annular means.

F.2 Effective decay rate

The transition product in equation 22 motivates quantifying how effective-kernel strength attenuates with distance. We therefore introduce an effective exponential decay rate α to summarize the measured radial profiles. For each sample n and branch b, we fit

$$\log d_b^{(n)}(r) \approx \beta_b^{(n)} - \alpha_b^{(n)} r, \tag{63}$$

by ordinary least squares. Larger α indicates faster radial attenuation.

As shown in Table 4, the spatial branch has a substantially larger decay rate than the spectral branch on both datasets, supporting their complementary roles in localized and domain-wide interactions. This contrast is more pronounced on FNS-KF, where faster spatial decay accompanies weaker spectral attenuation, consistent with the coexistence of local transport and nonlocal coupling in incompressible flow. For the spectral branch, α summarizes the overall attenuation trend rather than a strict exponential decay law.

F.3 Spatial branch variants

Figure 5(c) compares the spatial branches of separately trained one-, four-, and eight-directional variants. A single traversal produces a predominantly one-sided response, four traversals extend influence along both coordinate axes, and eight additionally strengthen diagonal interactions. The comparison shows that additional scan directions broaden spatial coverage, although the kernels remain direction-dependent.